A MACRO THAT PREVENTS COLUMN OVERWRITING

Brian L. Bournival, MedFocus, Lake Bluff, IL
Henry B. Winsor, MedFocus, Lake Bluff, IL

ABSTRACT

We have developed a macro called OVERWRIT that tests whether all values of a character variable will fit within the space designated for it. If not enough or too much space is allocated, the macro provides useful information to quickly correct the problem.

INTRODUCTION

When data is output in tabular form, columns will overlap if the length of a variable value exceeds the space allocated for it. SAS® does not issue a warning or error message when this occurs, and when large datasets are processed, it can be very difficult to detect. One possible solution to the problem is to allocate a large number of spaces for every character variable of indeterminate length. However, this may not be practical, and could result in many more pages of output than would be necessary if the optimum number of spaces were allocated for each character variable. We believe a better approach would be to use the OVERWRIT macro described in this paper.

MACRO DESCRIPTION

The central concept of OVERWRIT is fairly simple. Within a DATA _NULL_ step, the maximum length of a character variable is calculated and compared to the number of spaces allocated for it. If any of the variable values do not fit within the column boundaries, the macro calculates the number of additional spaces required, and issues an error message to the SAS log with the following information:

1. The variable value with the most characters.
2. The number of characters in the longest value.
3. The number of spaces available for the variable.
4. The beginning and ending column locations and the number of spaces between columns.
5. The number of additional spaces required to fit all values of the variable within the column boundaries.

If all values do fit, OVERWRIT prints the similar information as when overwriting occurs, including the number of spaces in excess of what is necessary to print the longest value LISTVAR. The macro must be called outside a dataset.

PARAMETER DESCRIPTIONS AND VALIDATIONS

OVERWRIT has seven parameters, of which are required. The main required parameter is LISTVAR, the variable to be tested. The macro tests LISTVAR to ensure that it exists and that it is a character variable.

In order to do this, an array composed of character variables in the dataset is created. The name of each element in the array is determined using CALL VNAME, and each name is compared to LISTVAR.

```
LENGTH VNAME $8;
ARRAY CHAR [*] CHARACTER_
CHARFLG=0;
DO I=1 TO DIM(CHAR) WHILE(CHARFLG=0);
   CALL VNAME(CHAR(I),VNAME);
   IF "&LISTVAR"=VNAME THEN CHARFLG=1;
END;
```
If LISTVAR is found in the character variables array, the macro proceeds. However, if it is not found, an error condition exists because OVERWRIT can only accurately determine the lengths of character variables. To further define the error, LISTVAR is compared to elements in a numeric variables array. If it is found, it is identified as a numeric variable, whereas, if it is not found, it is identified as not existing in the dataset.

The other required parameter is the beginning column location, BEG. SPACING is the number of spaces between columns; it defaults to 2. The ending column location, END, is actually the beginning of the next column; END defaults to the beginning of the column after the end of the line (133+SPACING, assuming a maximum line length of 132). BEG, END, and SPACING are all tested with the DATATYPE function to ensure that they are numeric entries.

```
%IF %DATATYPE(&BEG)^=NUMERIC %THEN %DO;
   PUT 'ERR' 'OR: BEG MUST BE NUMERIC.';
   ABORT RETURN;
RUN;
%END;
```

The RUN statement above forces data step execution at this point, and prevents further compiler processing.

The parameters are also tested to ensure that they fall within acceptable boundaries.

```
%IF %EVAL(&BEG)<1 | %EVAL(&BEG)>132 %THEN %DO;
   PUT 'ERR' 'OR: BEG MUST BE BETWEEN 1 and 132.';
   ABORT RETURN;
RUN;
%END;
```

The data type test must be conducted in a step prior to the boundary limitations test, because an erroneous character value for BEG will result in a system error in the %EVAL function of the boundary limitations test.

DLIB and DSET represent the data library and dataset that contain the variable to be tested. Both parameters are optional with default to WORK and _LAST_., respectively.

```
DATA _NULL_; SET &DLIB..&DSET END=EOF;
```

The value of the optional HDR parameter is the longest line of the column heading. If included in the macro call, this line will also be tested for whether it will fit within the column boundaries.

**MACRO EXECUTION**

If all parameter values are valid, macro goes to work. First, the number of spaces allocated for LISTVAR is calculated.

```
ALLOCATE=%EVAL(&END-&BEG-&SPACING);
```

Both BEG and SPACING must be subtracted from END, because END represents the beginning of the next column, not the end of the current column.

Then the length and value of the long LISTVAR value is determined.

```
LENGTH MAXVAL $200;
RETAIN MAXLEN MAXVAL;
IF LENGTH(&LISTVAR)>MAXLEN THEN DO;
   MAXLEN=LENGTH(&LISTVAR);
   MAXVAL=\"\"||TRIM(&LISTVAR)||\"\";
END;
```

The rest of the macro executes at end-of-MAXLEN is compared to the length of HDR, longest line of the column heading. If HDR is longer than MAXLEN, its length and value become the maximum values in our test.

```
IF EOF & %LENGTH(%STR(&HDR))>MAXLEN THEN DO;
   MAXLEN=%LENGTH(%STR(&HDR));
   MAXVAL=\"\"||\"QUOTE(&HDR)\"||\"\"||\"(THE HEADING)\";
END;
```
We include the label "THE HEADING" in MAXVAL to differentiate it from a LISTVAR value. This can be useful because the column heading may be more easily changed than a LISTVAR value.

The last step of the macro is to print the appropriate message to the SAS log (see MACRO DESCRIPTION).

**SAMPLE APPLICATIONS**

1. If the longest variable value (including the column heading) is less than the space allocated for it, OVERWRIT prints to the SAS log a message similar to the one below. In this example, three spaces could be allocated to another column without overwriting occurring.

   ```sas
   %OVERWRIT(listvar=tooshort, hdr=heading, beg=1, end=12, spacing=1);
   ```

   ```sas
   longest value of tooshort = "heading" (the heading)
   number of characters in longest value of tooshort = 7
   number of available spaces for tooshort = 10
   you have allocated 3 more spaces than are necessary for tooshort.
   begin column = 1 / end column = 12 / space between columns = 1
   ```

   The last two lines are omitted from the message if the maximum length of the variable is equal to the space allocated for it.

2. If the longest variable value is larger than its designated column boundaries, an error message similar to the one below is printed to the SAS log. In this case, the longest value of TOOLONG, "OPEN 2", will not fit within the column; if TOOLONG is not modified, two additional spaces must be allocated for it.

   ```sas
   %OVERWRIT(listvar=toolong, beg=12, end=19);
   ```

   ```sas
   error: the longest value of toolong has too many characters to fit into the space allocated for it.
   ```

**CONCLUSION**

When large datasets need to be output in tabular form, efficient utilization of space is very desirable. With the information provided OVERWRIT, column spacing can easily be optimized so that character variable values can be placed within their column boundaries, and column widths are only as large as they have to be.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. © indicates USA registration.